LINEAR MODELS FOR CLASSIFICATION (PART 1)

Hsin-Min Lu
盧信銘
台大資管系

Topics

- Introduction
- Discriminant Function
- Probabilistic Discriminative Models
- Laplace Approximation
- Bayesian Logistic Regression

INTRODUCTION

Introduction

- Regression:
  - Map input vector x to one or more continuous target variables t
- Classification:
  - Map input vector x to one of K discrete classes $(K \geq 2)$
- Ordinal Regression:
  - Map input vector x to one of K discrete classes that have an ordering (e.g., on-line evaluation (stars))

Linear Classification Models

- Simple scenario: disjoint classes
- Divide input space into decision regions
- Linear model decision surface are linear function of input x
  - D-1 dimension hyperplane in D dimensional input space
- Linearly separable:
  - Classes in a dataset can be separated exactly using linear decision surface

Probabilistic Models

- Converting Regression to Class Output
- Target variables *t* now represent class labels
- Binary class representation is convenient
  - Two classes: $t \in \{0,1\}$, t=1 represent C_1; t=0 represent C_0
  - Interpret value of *t* as the probability that class is C_1
  - Probabilities need to take value between 0 and 1
  - For K>2, use 1-of-K coding scheme
    - E.g. K=5, $t = (0,1,0,0,0)^T$ represents class 2
    - Value t_k interpreted as probability of class C_k

Two Approaches to Classification

- Discriminant function
  - Directly assign x to a specific class
  - E.g. Fisher's linear discriminant function, perceptron
- Probabilistic models
  - Model $p(C_k|x)$ and make optimal decisions

- Probabilistic models separating inference from decision
  - Accommodate different loss function
  - Compensate for unbalanced data
  - Combine models

Probabilistic Classification Models

- Generative
  - Model class conditional densities by $p(x|C_k)$ together with prior probabilities $p(C_k)$
  - $p(C_k|x) \propto p(x|C_k)p(C_k)$

- Discriminative
  - Directly model conditional probabilities $p(C_k|x)$

Converting Linear Regression Models to Linear Classification Models

- Linear regression model y(x,w) is a linear function of parameters w
- In simplest case also a linear function of variable x
  - $y(x) = w^T x + w_0$
- For classification we wish to obtain a discrete output or posterior probabilities in range (0,1)
- Use generalization of this model $y(x) = f(w^T x + w_0)$
  - $f(\cdot)$ is an activation function
  - $f^{-1}(\cdot)$ is a link function

Generalized Linear Model

- $y(x) = f(w^T x + w_0)$
- The decision surface correspond to $y(x) = constant$ or $w^T x + w_0$=constant
- So decision boundaries are linear even if $f(\cdot)$ is nonlinear
  - ➔Generalized Linear Model
- No longer linear w.r.t to w due to $f(\cdot)$
  - Lead to more complex models for classification than regression

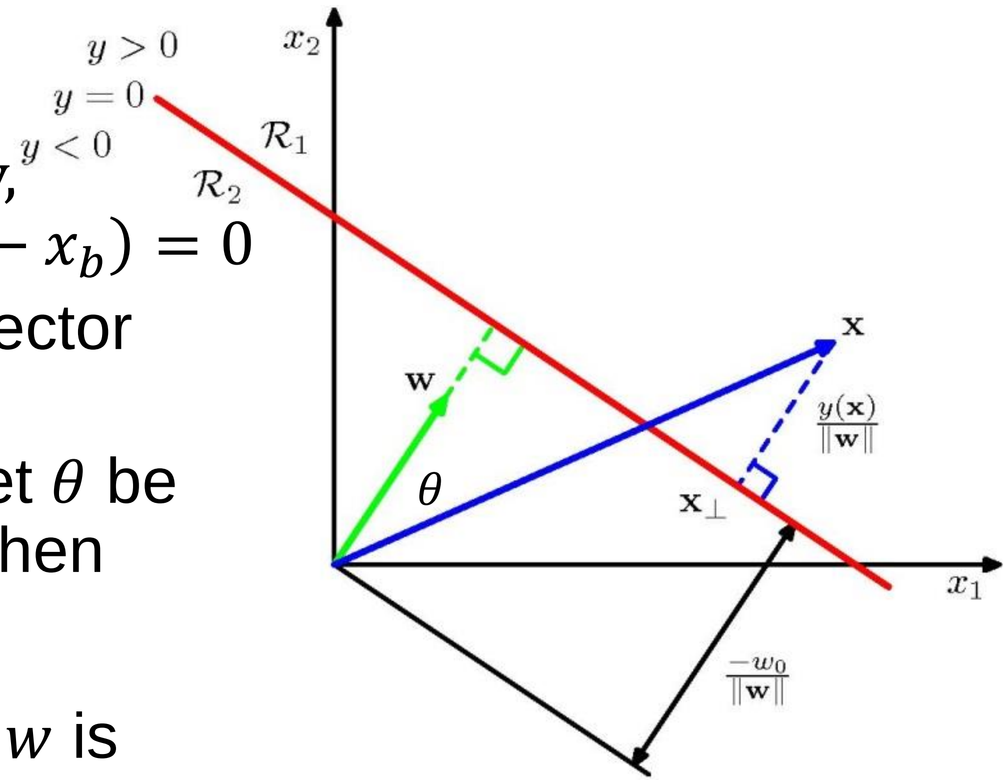
DISCRIMINANT FUNCTIONS

Linear Discriminant Function

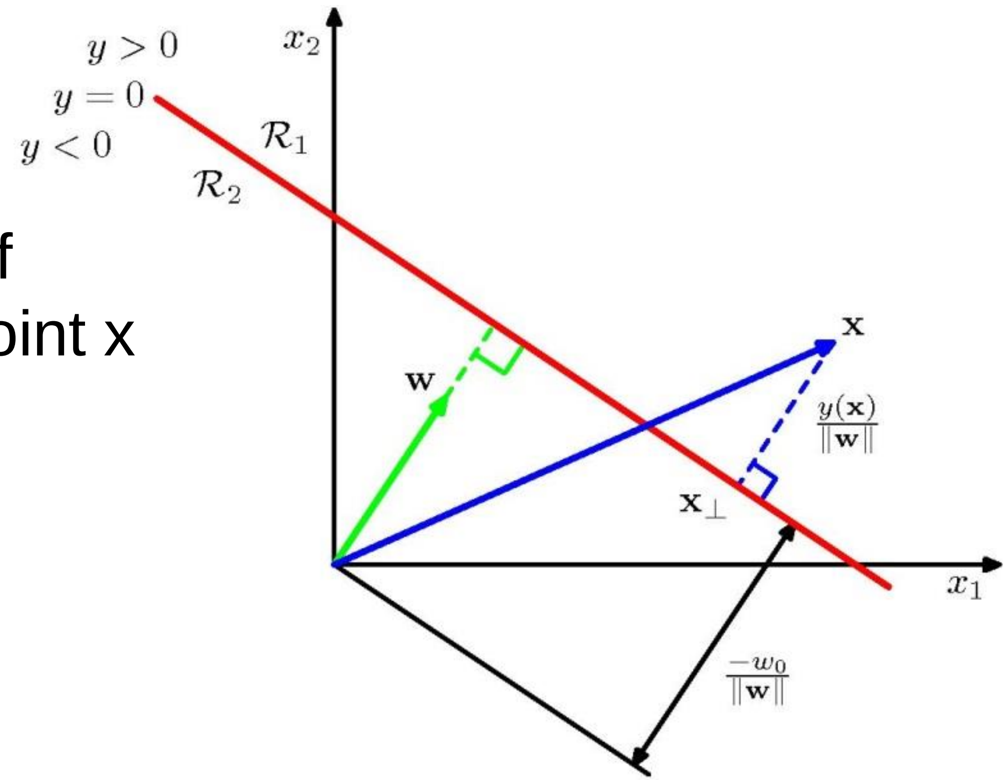
- Assigns input vector x to one of K classes denoted by C_k
- Restrict attention to linear discriminants (decision hyperplanes)
- Two-class linear discriminant function:
  - $y(x) = w^T x + w_0$
  - w is a weight vector and w_0 is bias
  - Assign x to C_1 if $y(x) \geq 0$, else C_2

Geometry of Linear Discriminant Functions

- $y(x) = w^T x + w_0$
- Decision boundary: $y(x) = 0$
- For x_a and x_b on the boundary, $y(x_a) - y(x_b) = 0$, ➔ $w^T(x_a - x_b) = 0$
- ➔ w is orthogonal to every vector on surface.
- Given an arbitrary vector x, let θ be the angle between x and w, then $\cos\theta = \frac{x \cdot w}{\|x\|\|w\|}$
- The projection length of x on w is $\|x\|\cos\theta = \frac{w^T x}{\|w\|} = \frac{y(x) - w_0}{\|w\|} = \frac{y(x)}{\|w\|} + \frac{-w_0}{\|w\|}$
- If x is on the surface, then $y(x) = 0$, the distance between origin and x is $\frac{-w_0}{\|w\|}$



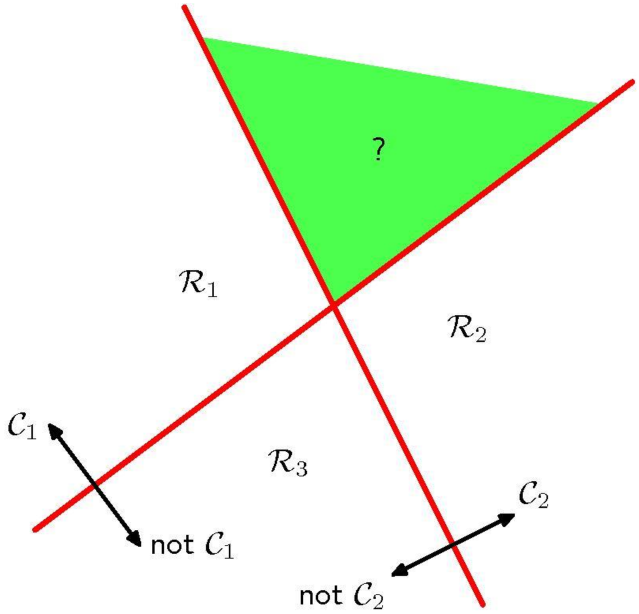
Geometry of Linear Discriminant Functions



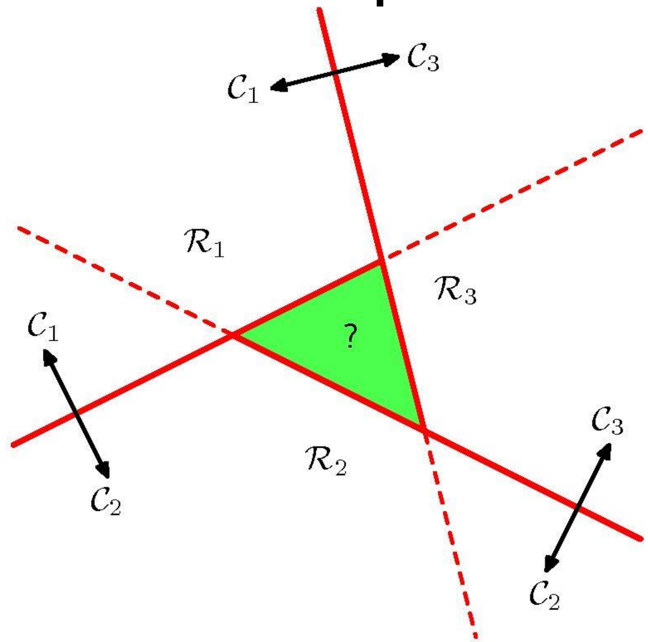
- $y(x) = w^T x + w_0$
- $y(x)$ gives signed measure of perpendicular distance r of point x to decision surface ($r = \frac{y(x)}{\|w\|}$)
- w_0 sets distance of origin to surface ($\frac{y(0)}{\|w\|} = \frac{w_0}{\|w\|}$)
- With dummy input $x_0 = 1$, and $\omega = (w_0, w^T)^T$, $z = (x_0, x^T)^T$, then $y(x) = \omega^T z$
  - Pass through origin in augmented D+1 dimensional space

Multiple Classes with 2-class classifiers

- One-versus-the-rest
- Build a K class discriminant using K-1 classifiers



- One-versus-one
- Build K(K-1)/2 binary discriminant classifiers, one for each pair



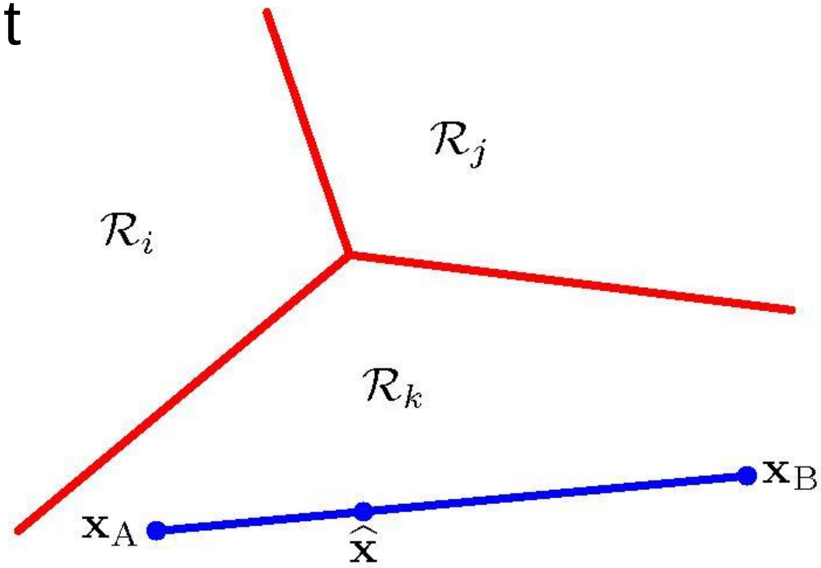
Both result in ambiguous regions of input space

Multiple Classes with K Discriminants

- Consider a single K class discriminants of the form
  - $y_k(x) = w_k^T x + w_{k0},\ k = 1,2,\ldots,K$
  - Assign a point x to class C_k if $y_k(x) > y_j(x)$ for all $j \neq k$
- Decision boundary between class C_k and C_j is given by $y_k(x) = y_j(x)$
  - Corresponds to D-1 dimensional hyperplane defined by
  $$\left(w_k - w_j\right)^T x + \left(w_{k0} - w_{j0}\right) = 0$$
  - Same form as the decision boundary for 2-class case
  - Decision regions of such a discriminant are always singly connected and convex
    - Proof on next slide

Convexity of Decision Regions

- Consider two points x_a and x_b both in decision region R_k
- Any point $\hat{x}$ on line connecting x_a and x_b can be expressed as $\hat{x} = \lambda x_a + (1 - \lambda)x_b$, where $0 \leq \lambda \leq 1$.
- From linearity of discriminant functions $y_k(x) = w_k^T x + w_{k0}$
- Combining the two, we have $y_k(\hat{x}) = \lambda y_k(x_a) + (1 - \lambda) y_k(x_b)$
- Because x_a and x_b lie inside R_k, it follows that $y_k(x_a) > y_j(x_a)$ and $y_k(x_b) > y_j(x_b)$ for all $j \neq k$.
- Hence $\hat{x}$ also lies inside R_k
- Thus R_k is singly-connected and convex



Learning the Parameters of Linear Discriminant Functions

- Three methods
  - Least squares
  - Fisher’s Linear Discriminant (omitted)
  - Perceptrons

Least Square for Classification

- Simple closed-form solution exists
- Each C_k, $k = 1,2, \ldots, K$, is described by its own linear model

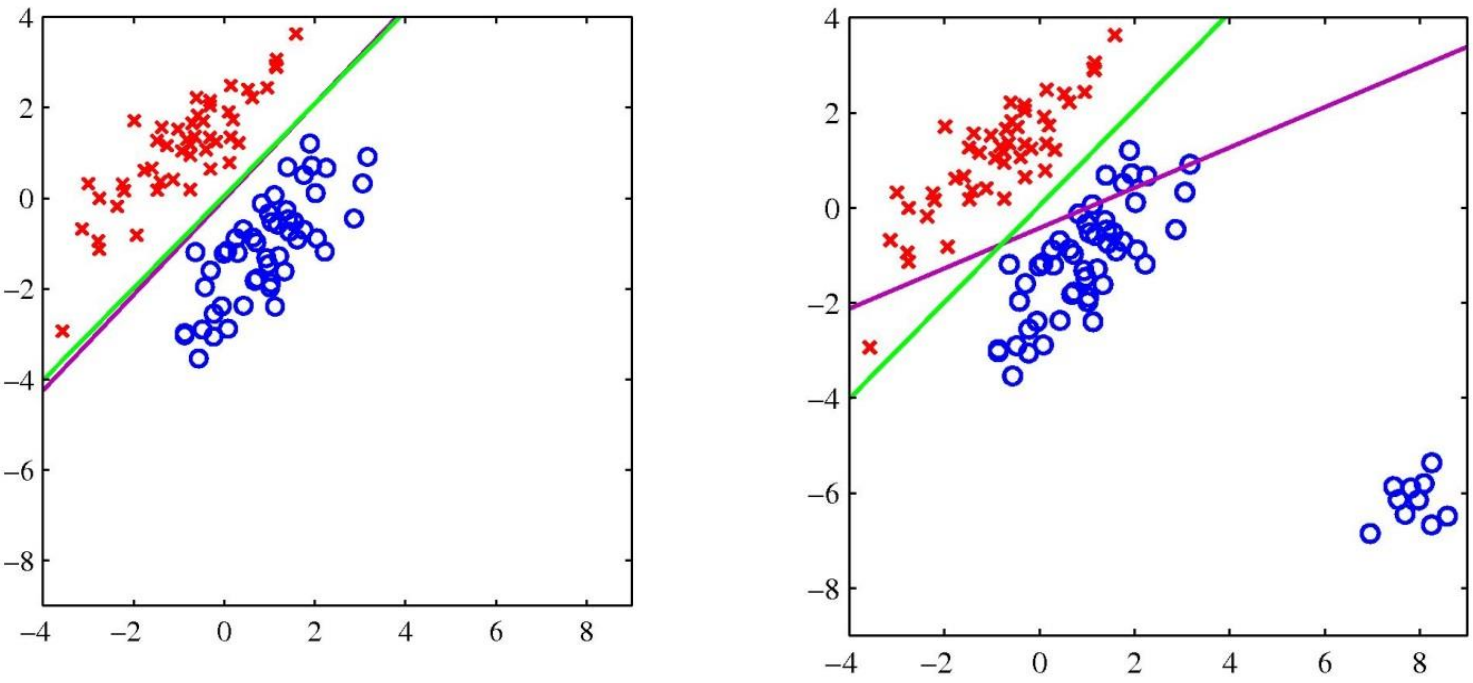
$$y_k(x) = \widetilde{w}_k^T x + w_{k0}$$

- Create augmented vector $x = (1, x^T)^T$ and $w_k = \left(w_{k0}, \widetilde{w}_k^T\right)^T$ (length is D+1)
- There are k parameter vectors: $W = [w_1\ w_2\ \ldots w_K]$
- For input feature x, score for class k: $y_k(x) = w_k^T x$
- Input x is assigned to class for which output $y_k = w_k^T x$ is largest.
- Learn W by minimizing squared error.

Learning Parameters Using Least Squares

- Training data $\{x_n, t_n\}$, $n = 1,2, \dots, N$.
  - t_n is a column vector of K dimensions using 1-of-K form.
  - x_n is also a column vector of length D+1.
- Learn w_k for each class separately.
- For class k, collect $\{t_{n,k}\}$, the k-th element in t_n for $n = 1, 2, \dots, N$; form a column vector T_k using $\{t_{n,k}\}$
- Form matrix X by stacking x_n^T
- $\widehat{w}_k = [X^T X]^{-1} X^T T_k$.
- Do this for all k.

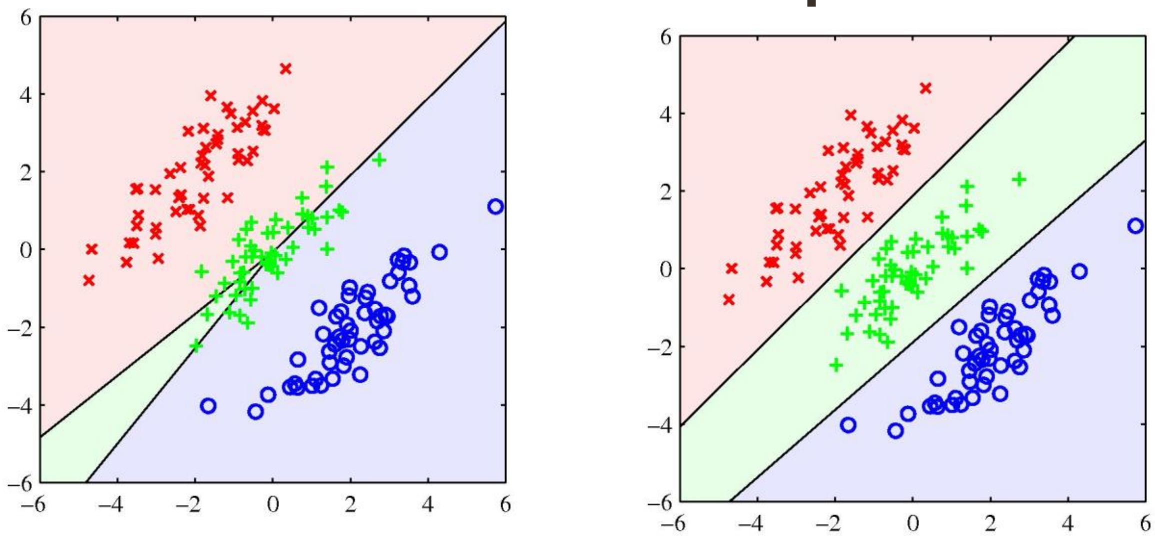
[Limitation] Least Square is Sensitive to Outliers



- Sum of squared errors penalized predictions that are “too correct” (magenta lines)
  - Long way from decision boundary
  - Logistic regression (green lines) does not have this problem
  - SVMs have an hinge error function that have a similar shape to that of the logistic regression

Disadvantages of Least Squares

Ordinary regression



Logistic regression

- Regions assigned to green class is too small, mostly misclassified
- Logistic regression performs well
- Disadvantage of least square:
  - Lack robustness to outliers
  - Gaussian assumption may not be a good choice for classification problems
  - Binary target values have a distribution far from Gaussian

Perceptron Algorithm

- Two-class model
  - Input vector x transformed by a fixed nonlinear function to give feature vector $\phi(x)$
  - $y(x) = f(w^T\phi(x))$
    where non-linear activation $f(\cdot)$ is a step function
  - $f(a) = \begin{cases} +1 \; if \; a \geq 0 \\ -1 \; if \; a < 0 \end{cases}$
- Use a target coding scheme
  - t=1 for class C_1, and t=-1 for C_2 similar to the activation function

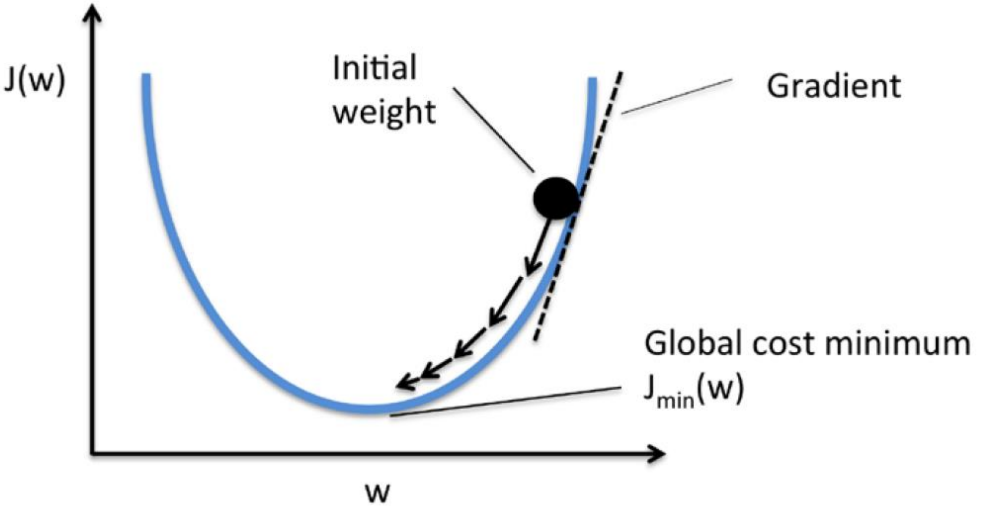
Perceptron Error Function

- Error function: number of misclassifications
- This error function is a piecewise constant function of w with discontinuities (unlike regression)
- Hence no closed form solution

Perceptron Criterion

- Seek w such that $x_n \in C_1$ will have $w^T x_n \geq 0$ where as $x_n \in C_2$ will have $w^T x_n < 0$
- Target value $t \in \{-1, 1\}$. C_1 ➔ t=1, all patterns need to satisfy $w^T \phi(x_n) t_n > 0$
- For each misclassified sample, perceptron criterion tried to minimize $E_p(w) = -\sum_{n \in \mathrm{M}} w^T \phi_n t_n$
- M denotes set of all misclassified patterns and $\phi_n = \phi(x_n)$

Learning by Gradient Descent



- Loss function $J(\boldsymbol{\theta}) = \sum_{i=1}^{m} L(\boldsymbol{x_i}, \boldsymbol{y_i}, \boldsymbol{\theta})$
- Search for the best $\boldsymbol{\theta}$ by minimizing the loss.
- **Gradient descent:**
- At step t with parameter $\boldsymbol{\theta}^t$, we can find $\boldsymbol{\theta}^{t+1}$ that (hopefully) reduces the loss function by setting

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \epsilon \boldsymbol{g}$$

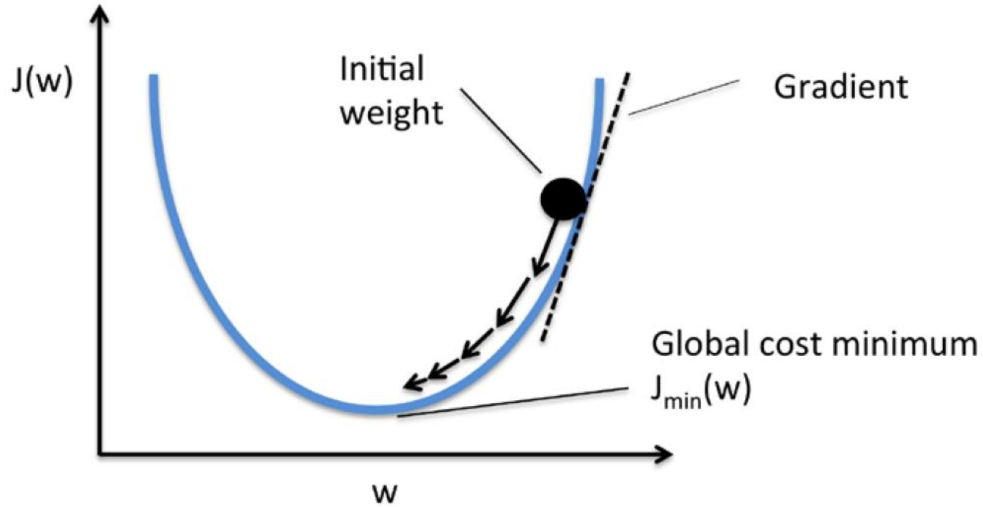
- $\boldsymbol{g} = \nabla_{\boldsymbol{\theta}} \mathrm{J}(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_1} \\ \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_2} \\ \vdots \\ \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_K} \end{bmatrix}$ is the gradient of the loss function
- ϵ is the learning rate (a small positive real value).

Learning by Stochastic Gradient Descent (SGD)

- When computing gradient $\boldsymbol{g} = \nabla_{\boldsymbol{\theta}} \mathrm{J}(\boldsymbol{\theta})$, we need to sum over all training instances
- $\boldsymbol{g} = \nabla_{\boldsymbol{\theta}} \mathrm{J}(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \nabla_{\theta} L(\boldsymbol{x_i}, \boldsymbol{y_i}, \boldsymbol{\theta})$.
- This will take a long time if we have a large number of training instance.
- Instead, we can sample a minibatch of examples $B = \left\{x^{(1)}, x^{(2)}, \dots, x^{(m')}\right\}$ drawn randomly from the training set.
- Estimate the gradient by $\boldsymbol{g} = \frac{1}{m'} \sum_{i=1}^{m} \nabla_{\theta} L\left(\boldsymbol{x}^{(\boldsymbol{i})}, \boldsymbol{y}^{(\boldsymbol{i})}, \boldsymbol{\theta}\right)$.
- Update the parameter by $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} - \epsilon \boldsymbol{g}$

SGD Summary

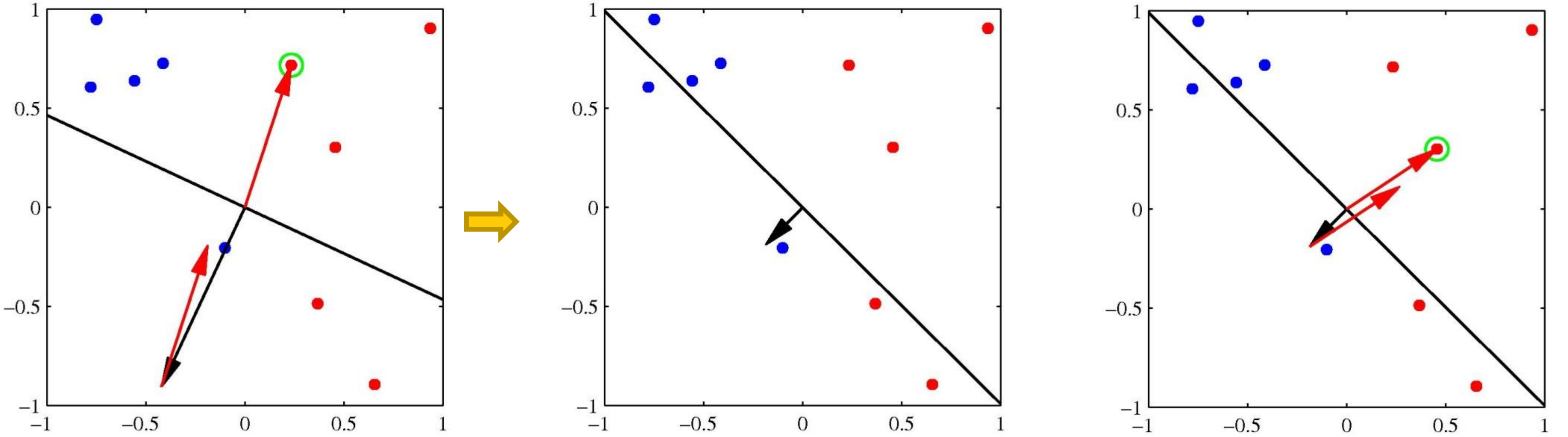
- Loss function $J(\boldsymbol{\theta}) = \sum_{i=1}^{m} L(\boldsymbol{x_i}, \boldsymbol{y_i}, \boldsymbol{\theta})$
- Draw a minibatch of m' training instances
- Estimate the gradient using the minibatch $\boldsymbol{g} = \frac{1}{m'} \sum_{i=1}^{m} \nabla_\theta L\left(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}, \boldsymbol{\theta}\right)$.
- Update the parameter by $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} - \epsilon \boldsymbol{g}$

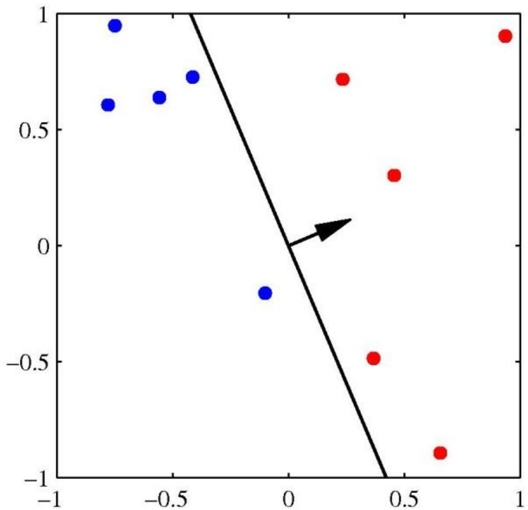


Perceptron Algorithm

- Error function $E_p(w) = -\sum_{n\in \mathrm{M}} w^T \phi_n t_n \equiv \sum_{n\in M} E_n$
- **Stochastic gradient descent**
  - a.k.a Sequential gradient descent
  - Loop through M, for each data point n, change in weight is given by
  - $w^{\tau+1} = w^{\tau} - \eta \nabla \mathrm{E_n(w)} = w^{\tau} + \eta \phi_n t_n$
  - $\nabla E_n(w) = \frac{\partial E_n(w)}{\partial w} = -\phi_n t_n$
    η is learning rate (set to 1), τ is a step index
- The algorithm
  - Cycle through the training patterns in turn
  - If incorrectly classified for class C_1, add to weight vector
  - If incorrectly classified for class C_2, subtract from weight vector

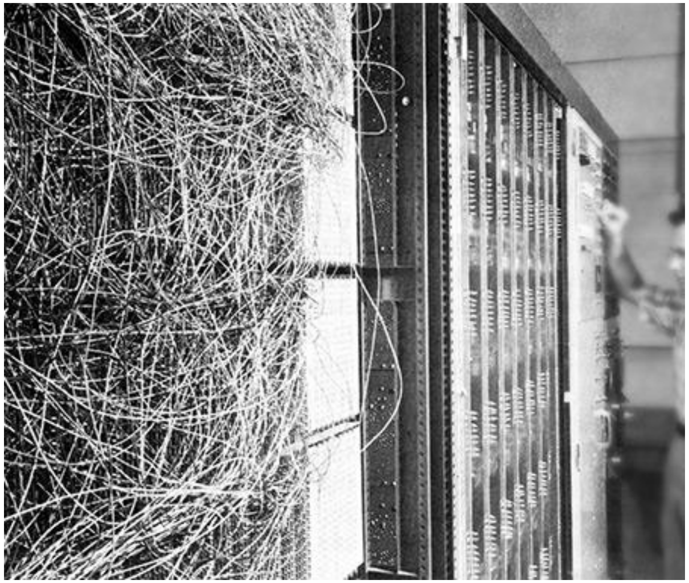
Perceptron Learning





- Green point is misclassified ➔ add to weight vector (black arrow)

History of Perceptron



- Perceptron invented by Frank Rosenblatt
- Mark 1 perceptron hardware (patching board, allowing different configurations)

Disadvantages of Perceptrons

- Does not converge if classes not linearly separable
- Does not provide probabilistic output
- Not readily generalized to K>2 classes

Comparison of Parameter Learning Approaches

- Least Squares
  - Not robust to outliers, model close to target values
- Fisher's linear discriminant (omitted)
  - Separation determined by within-class and between class variance
  - Can be seen as a special case of least square
- Perceptrons
  - Does not converge if classes not linearly separable
  - Does not provide probabilistic output
  - Not readily generalized to K>2 classes

QUESTIONS?